response to receiving the quiesce event, the second thread determines whether it is holding any resource required by another thread, quiesces only if it determines that it is not holding any resource required by another thread, and upon quiescing resumes execution of the first thread to process the application event.

Claims 1 and 12 as amended and the claims dependent thereon are believed to distinguish patentably over the prior art, including Jackson 5,297,274 (Jackson) and Anschuetz et al. 5,305,455 (Anschuetz).

The Examiner contends that it would have been obvious to combine the teachings of Jackson with the per thread exception management of Anschuetz "for the purpose of the whole process not be suspended and notifying individual threads and allowing them a chance to terminate gracefully" (Paper no. 5, pp. 3-3). In point of fact, not only would it not have been obvious to combine the two references in the manner suggested by the Examiner, but even if the references were so combined, the combination would not result in applicants' claimed invention.

As for the motivation for combining the two references, Jackson relates to a system for analyzing the performance of software systems by suspending their execution in a controlled manner, whereas Anschuetz relates to a system for managing process termination[1] exceptions on a per thread basis in a multitasking system. These objectives are

---

[1]Contrary to the Examiner's repeated assertions on page 3 of the action, Anschuetz discloses only the termination, and not suspension, of threads on an individual basis.

completely orthogonal to each other, such that neither reference would be regarded as being relevant to the implementation of the other. They are hardly "in the same field of endeavor" as suggested by the Examiner (Paper no. 5, p. 3).

More importantly, even if it were regarded as obvious to modify the system of Jackson to include the per thread exception management of Anschuetz, there would still be lacking any suggestion of checking to determine whether a thread being quiesced is holding critical resources, as recited in Claims 1 and 12 as amended. As applicants have previously noted, the problem of resource deadlock (and hence, the motivation to solve it) does not arise in the system environments of Jackson and Anschuetz, where only one thread executes at a time (because the processor is a uniprocessor) and where a thread can only be interrupted to dispatch another thread when it has released any critical resources. Such resource deadlock can be a real problem, however, in a multiprocessor environment in which other threads may hold critical resources. Merely quiescing threads on an individual basis does not ensure that such threads have released resources required by other threads. Neither cited reference, either singly or in combination, suggests this aspect of applicants' claimed invention.

Respectfully submitted,

WAK:sk

William A. Kinnaman, Jr.
Registration No. 27,650
Telephone No. (914) 433-1175